Game Development for iPhone

Part 1

Session 304

These are confidential sessions—please refrain from streaming, blogging, or taking pictures.

Game Development for iPhone

Part 1

Allan Schaffer
Graphics Technology Evangelist

iPhone and iPod Touch

Fantastic platforms for handheld games

Technologies

Largest App Store Category

iPhone OS 3.0

Technologies for game development

iPhone 3G S

3G
9:42 AM
Tuesday
9
Messages
Calendar
Photos
Camera
YouTube
Stocks
Maps
280
73°
Weather
Voice Memos
Notes
Clock
Calculator
Settings
iTunes
App Store
Compass
Phone
Mail
Safari
iPod

Demo

Graeme Devine

Agenda

- Part 1
  - Welcome and Overview — 10 minutes
  - Graphics — 20 minutes
  - Audio — 30 minutes

- Part 2
  - Peer-to-Peer Connectivity — 20 minutes
  - Accelerometer — 20 minutes
  - In-App Purchase — 20 minutes

Agenda

- Part 1
  - Welcome and Overview — 10 minutes
  - Graphics — 20 minutes
  - Audio — 30 minutes
- Part 2
  - Peer-to-Peer Connectivity — 20 minutes
  - Accelerometer — 20 minutes
  - In-App Purchase — 20 minutes

Graphics

OpenGL|ES
TM

OpenGL ES

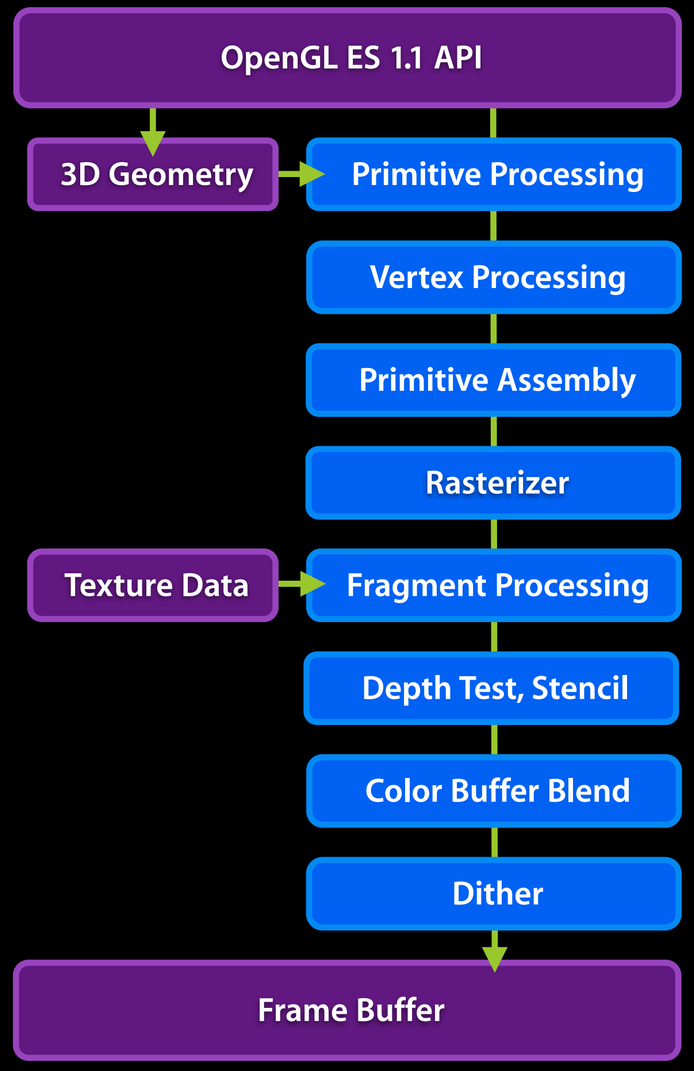
Overview

- High-performance 3D graphics API
  - The graphics API behind most games on iPhone
  - Designed for hardware-accelerated 3D rendering
- Mobile version of the OpenGL standard
  - Open industry standard
  - Distilled, compact, and efficient
  - Targeted to embedded devices

OpenGL ES Overview

OpenGL ES 1.1

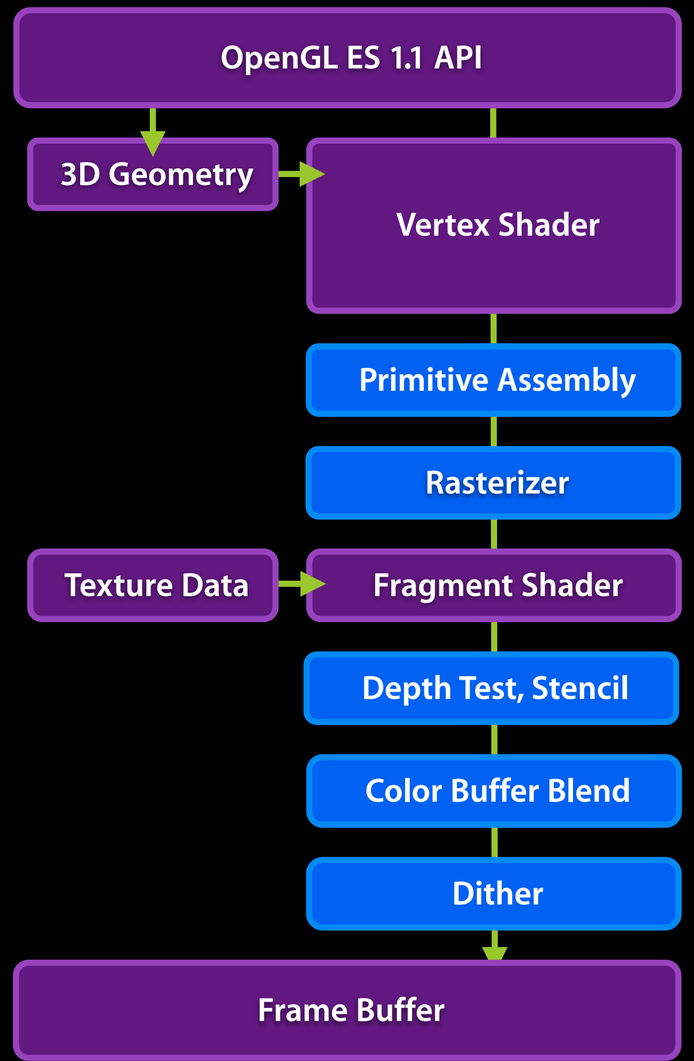
- Targets a *fixed-function* pipeline
- Fixed-function rendering steps:
  1. Create objects
     - Textures, buffers
  2. Set graphics state
     - Current texture, material, etc.
  3. Submit 3D geometry
     - Vertices, normals, colors, and texcoords



OpenGL ES Overview

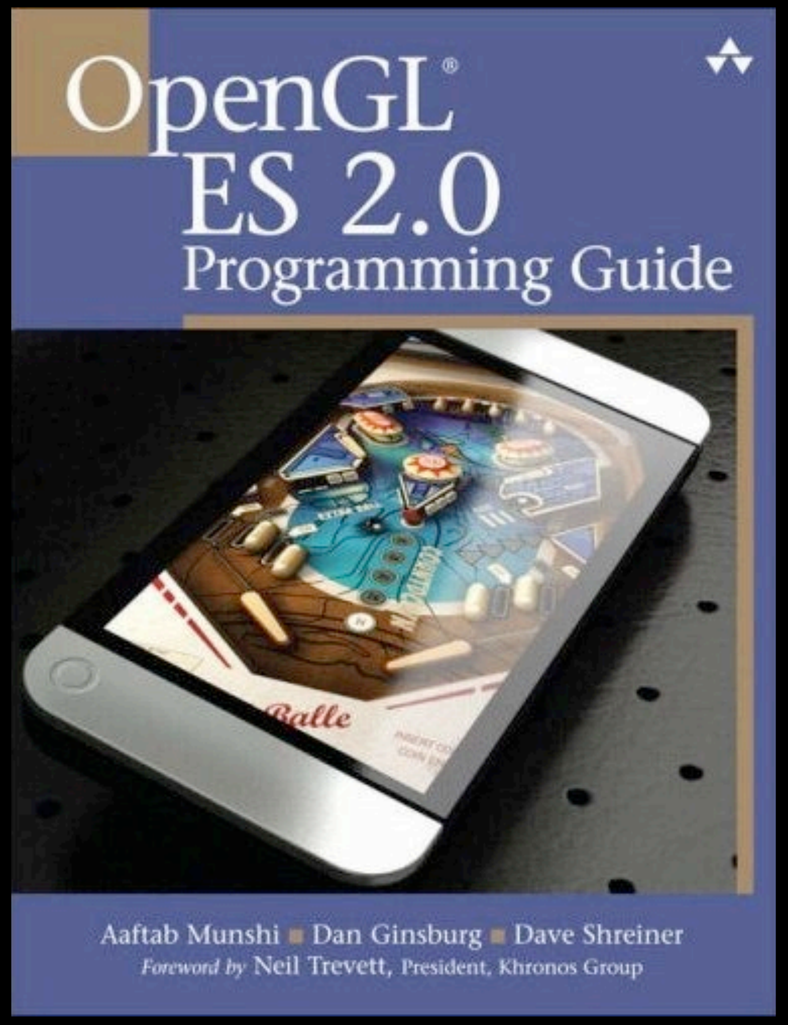
OpenGL ES 2.0

- Targets a *programmable* pipeline
- Programmable rendering steps:
  1. Create objects
     - Textures, buffers, shaders
  2. Set graphics state
     - Load shaders into GPU
  3. Submit 3D geometry
     - Uniforms, attributes, samplers



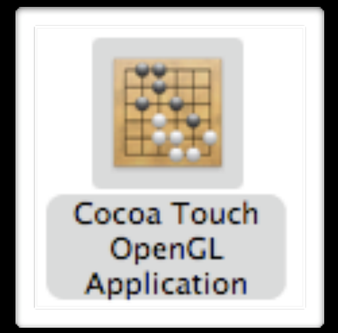
Learning the Basics

- OpenGL ES 2.0 Programming Guide
  - The "Gold Book" (it's purple)
- OpenGL Shading Language
  - The "Orange Book"
- http://developer.apple.com/iphone
  - OpenGL ES Programming Guide for iPhone
- OpenGL Programming Guide
  - The "Red Book"
- http://opengl.org
  - OpenGL ES specification, news, and forums



Learning the Basics

Getting started



- Cocoa Touch OpenGL ES Application Template
  - Starting point for an OpenGL ES-based application
  - EAGL setup, animation timer, and render callback
  - Implement -drawView: to draw a frame

EAGLView.m

EAGLView.m:90 -drawView

```
83
84      [EAGLContext setCurrentContext:context];
85
86      glBindFramebufferOES(GL_FRAMEBUFFER_OES, viewFramebuffer);
87      glViewport(0, 0, backingWidth, backingHeight);
88
89      glMatrixMode(GL_PROJECTION);
90      glLoadIdentity();
91      glOrthof(-1.0f, 1.0f, -1.5f, 1.5f, -1.0f, 1.0f);
92      glMatrixMode(GL_MODELVIEW);
93      glRotatef(3.0f, 0.0f, 0.0f, 1.0f);
94
95      glClearColor(0.5f, 0.5f, 0.5f, 1.0f);
96      glClear(GL_COLOR_BUFFER_BIT);
97
98      glVertexPointer(2, GL_FLOAT, 0, squareVertices);
99      glEnableClientState(GL_VERTEX_ARRAY);
100     glColorPointer(4, GL_UNSIGNED_BYTE, 0, squareColors);
101     glEnableClientState(GL_COLOR_ARRAY);
```

OpenGL ES on iPhone OS

OpenGL ES Platforms

Platform	OpenGL ES 1.1	OpenGL ES 2.0
iPhone 3G S	✓	✓
iPhone 3G	✓	✗
iPhone	✓	✗
iPod touch (2nd gen)	✓	✗
iPod touch	✓	✗

Both versions can coexist on the same device

OpenGL ES Platforms

Guidelines

- Choose OpenGL ES version based on requirements

Platform	OpenGL ES 1.1	OpenGL ES 2.0
iPhone 3G S	✓	✓
iPhone 3G	✓	✗
iPhone	✓	✗
iPod touch (2nd gen)	✓	✗
iPod touch	✓	✗

OpenGL ES Platforms

Guidelines

- Choose OpenGL ES version based on requirements
- All iPhone OS devices support OpenGL ES 1.1

Platform	OpenGL ES 1.1	OpenGL ES 2.0
iPhone 3G S	✓	✓
iPhone 3G	✓	✗
iPhone	✓	✗
iPod touch (2nd gen)	✓	✗
iPod touch	✓	✗

OpenGL ES Platforms

Guidelines

- Choose OpenGL ES version based on requirements
- All iPhone OS devices support OpenGL ES 1.1
- Games targeting iPhone 3G S capabilities can also provide an ES 1.1 fallback

Platform	OpenGL ES 1.1	OpenGL ES 2.0
iPhone 3G S	✓	✓
iPhone 3G	✓	✗
iPhone	✓	✗
iPod touch (2nd gen)	✓	✗
iPod touch	✓	✗

OpenGL ES Versions

Choosing your rendering API

- Configured by the EAGL context

```
EAGLContext* myContext =
    [[EAGLContext alloc] initWithAPI:kEAGLRenderingAPIOpenGLES2];
```

- or

```
EAGLContext* myContext =
    [[EAGLContext alloc] initWithAPI:kEAGLRenderingAPIOpenGLES1];
```

OpenGL ES Extensions

Extensions of interest

- OES_framebuffer_object
  - Destination for all rendering
  - Extension in 1.1, core in 2.0
- IMG_texture_compression_pvrtc
  - Saves memory
  - Hardware native
- APPLE_texture_2D_limited_npot
  - Non-power-of-two texture dimensions, e.g. 480 x 320
  - Supported for ES1.1 apps on iPhone 3G S

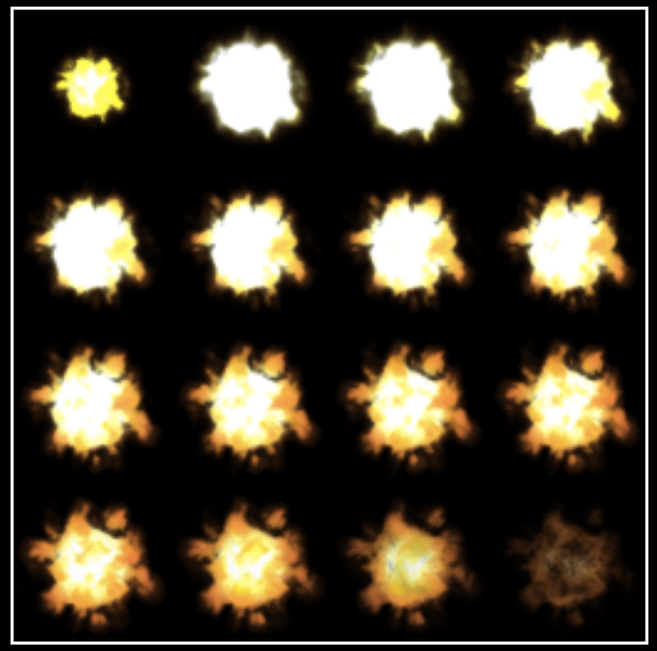
PVRTC

PowerVR Texture Compression

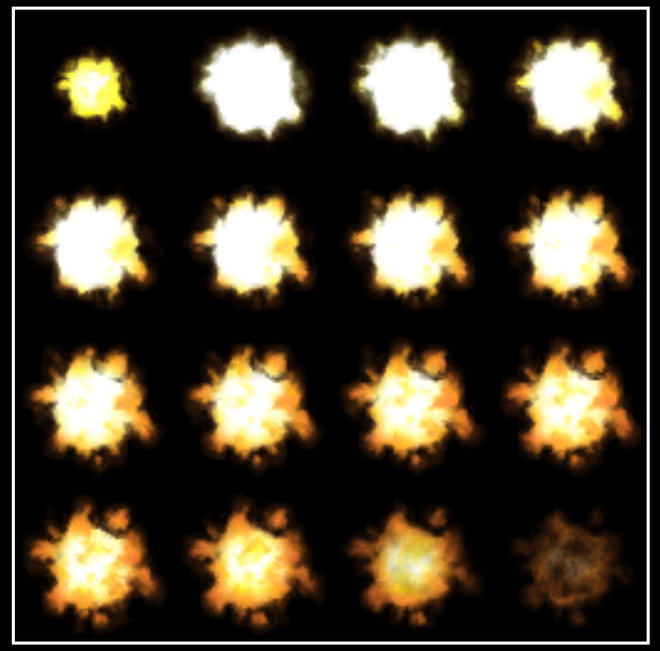
- Compress to 2 or 4 bits per pixel
  - Linear or perceptual weighting
- Saves memory
  - Reduces memory bandwidth pressure
- Compression tool provided

```
/Developer/Platforms/iPhoneOS.platform/Developer/usr/bin/texturetool
```

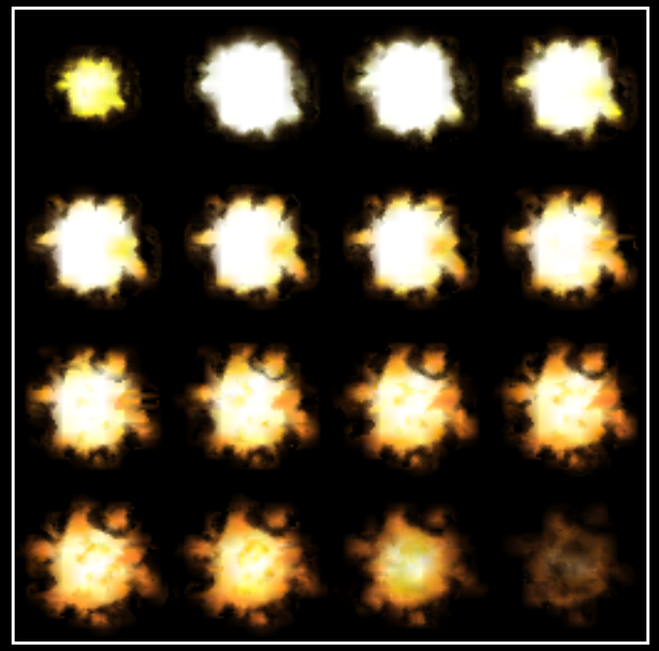
PVRTC



Uncompressed
32 bpp



Compressed
4 bpp



Compressed
2 bpp

Best Practices

Optimizing texture usage

- Minimize texture storage requirements
  - Use PVRTC (2- or 4-bit)
  - Use 16-bit (4444, 5551, 565) instead of 32-bit (8888)
  - Deallocate your copy of texture image data
- Minimize graphics state changes
  - Use texture atlases

One More Platform…

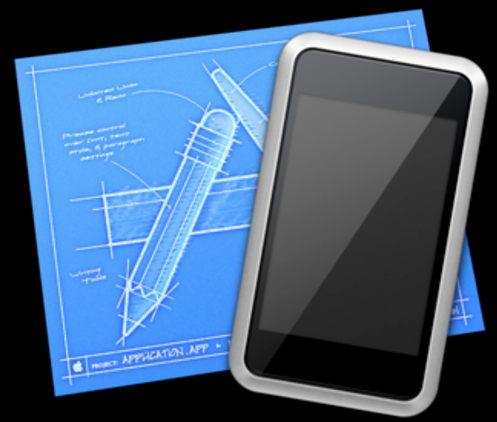
OpenGL ES Platforms

Platform	OpenGL ES 1.1	OpenGL ES 2.0
~~iPhone Simulator~~	✓	✓
~~iPhone 3G S~~		
iPhone 3G	✓	✗
iPhone	✓	✗
iPod touch (2nd gen)	✓	✗
iPod touch	✓	✗

iPhone Simulator

Full OpenGL ES 1.1 and OpenGL ES 2.0 implementation

- Compared to device
  - Functionality very similar
  - Performance very different
  - Faster turnaround time for experimentation
- Simulator currently does not enforce
  - Some OpenGL ES error generation
  - iPhone device resource limits
  - (Memory, scheduling, bandwidth, etc.)



Agenda

- Part 1
  - Welcome and Overview — Done
  - Graphics — Done
  - Audio — 30 minutes
- Part 2
  - Peer-to-Peer Connectivity — 20 minutes
  - Accelerometer — 20 minutes
  - In-App Purchase — 20 minutes

Audio

Core Audio

Audio Toolbox framework

- Broad API set for audio playback and recording on iPhone
  - Variety of APIs for different tasks
- For use in games or general-purpose applications

Core Audio

Game Audio

Scenario for today…

+

3D Game Audio

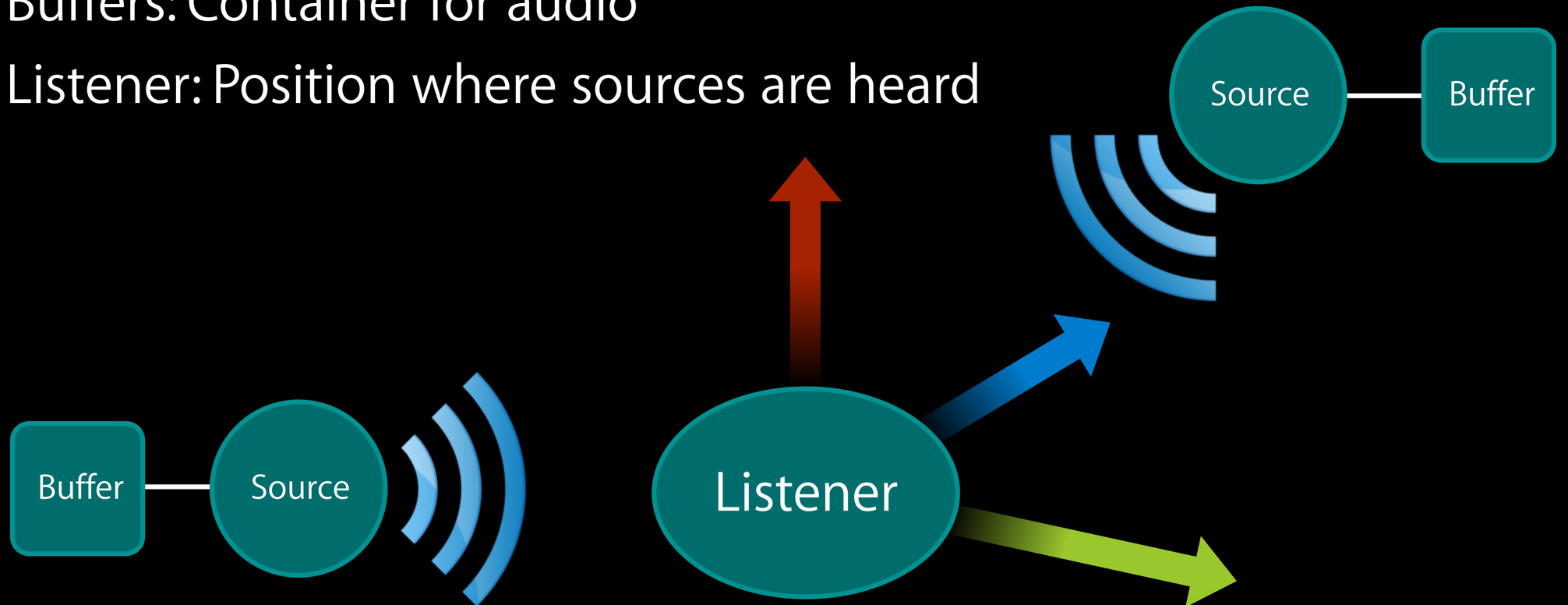
Game Audio with OpenAL

- Powerful API for 3D audio mixing
  - Especially for sound effects in games
- Models audio in 3D space, as heard by a single listener
  - Mimics OpenGL ES conventions
  - Same coordinate system
- iPhone implementation
  - OpenAL 1.1 without audio capture
  - Built atop I/O audio unit
    - Very low latency
- http://www.openal.org/

About OpenAL

Powerful environmental audio API

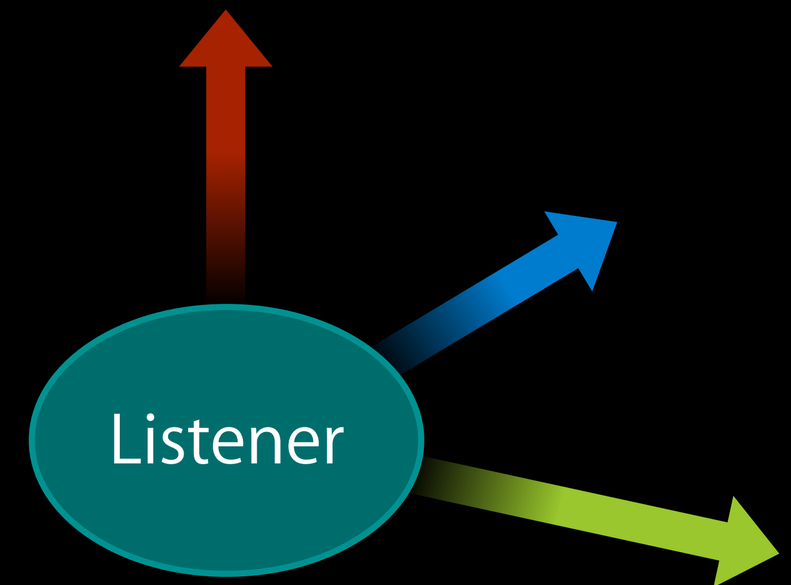
- Three fundamental concepts
  - Sources: 3D point emitting audio
  - Buffers: Container for audio
  - Listener: Position where sources are heard



Listener

OpenAL fundamentals

- Only one listener
- Positioned in the virtual space
- Orientation defined by two vectors
  - AT: Direction listener is facing
  - UP: Direction pointing “up”

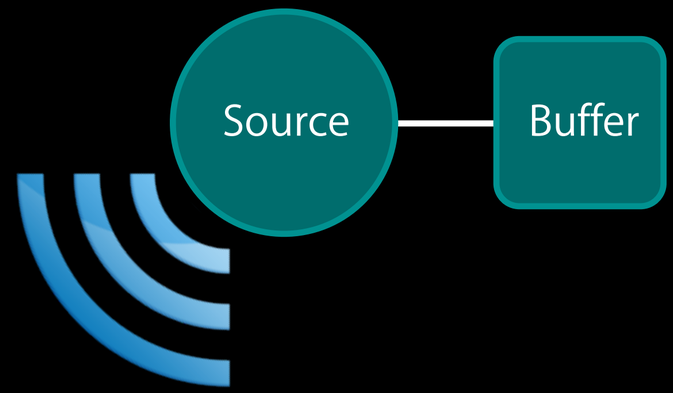



```
// Orient the listener facing +Z
ALfloat listenerOrientation[6] = { 0.0, 0.0, 1.0,     // AT vector +Z
                                   0.0, 1.0, 0.0 };   // UP vector +Y

alListenerfv(AL_ORIENTATION, listenerOrientation);
```

Sources

OpenAL fundamentals

- Each is a sound source in your game
- Positioned in the virtual space
- Many attributes to control rendering
  - Attenuation, pitch, looping, etc.

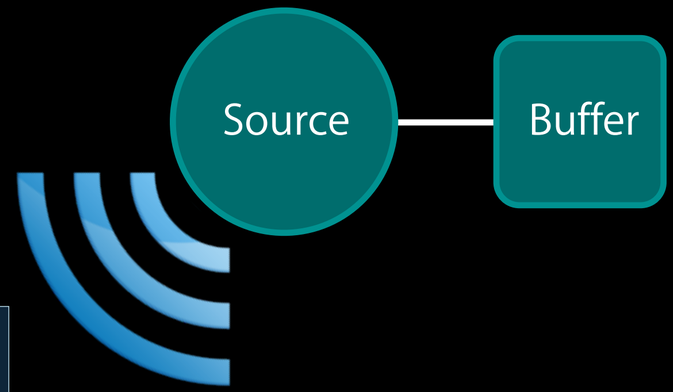



```
// Add a buffer to the source's queue
alSourcei (source, AL_BUFFER, buffer);

// Configure source attributes
alSourcei (source, AL_LOOPING, AL_TRUE);
alSourcefv(source, AL_POSITION, source_position_xyz);
alSourcef (source, AL_REFERENCE_DISTANCE, 5.0f);

// Start playing
alSourcePlay(source);
```

Buffers

Containers of audio data

- alBufferData()
  - OpenAL copies data to internal buffers




```
// Attach audio data to an OpenAL Buffer
alBufferData(buffer, format, data, size, freq);

// Release the audio data - OpenAL made a copy
releaseAudioData(data);
```

- alBufferDataStatic()
  - OpenAL uses memory provided by application
  - Application MUST NOT dispose data memory while in use

```
// Attach audio data to an OpenAL Buffer
// You must not release ‘data’ while in use.
alBufferDataStatic(buffer, format, data, size, freq);
```

Setting up OpenAL

Creating the listener

1

```
// Open an OpenAL Device
// Uses default system output device
device = alcOpenDevice(NULL);
```

2

```
// Create a new OpenAL Context (the mixer) for rendering
// Listener is implicit within the context
context = alcCreateContext(device, 0);
```

3

```
// Make the new context the Current OpenAL Context
alcMakeContextCurrent(context);
```

Setting up OpenAL

Creating the listener

1

```
// Open an OpenAL Device
// Uses default system output device
device = alcOpenDevice(NULL);
```

2

```
// Create a new OpenAL Context (the mixer) for rendering
// Listener is implicit within the context
context = alcCreateContext(device, 0);
```

3

```
// Make the new context the Current OpenAL Context
alcMakeContextCurrent(context);
```

Setting up OpenAL

Creating the listener

1

```
// Open an OpenAL Device
// Uses default system output device
device = alcOpenDevice(NULL);
```

2

```
// Create a new OpenAL Context (the mixer) for rendering
// Listener is implicit within the context
context = alcCreateContext(device, 0);
```

3

```
// Make the new context the Current OpenAL Context
alcMakeContextCurrent(context);
```

Setting up OpenAL

Creating sources and buffers

4
```
// Create an OpenAL Buffer Object to store audio data
alGenBuffers(1, &buffer);
```

5
```
// Fill the buffer with audio data
// Static buffer extension eliminates a buffer data copy
alBufferDataStatic(buffer, AL_FORMAT_MONO16, data, 4096, 44100);
```

6
```
// Create an OpenAL Source Object
alGenSources(1, &source);
```

7
```
// attach OpenAL Buffer to OpenAL Source
alSourcei(source, AL_BUFFER, buffer);
```

Setting up OpenAL

Creating sources and buffers

4
```
// Create an OpenAL Buffer Object to store audio data
alGenBuffers(1, &buffer);
```

5
```
// Fill the buffer with audio data
// Static buffer extension eliminates a buffer data copy
alBufferDataStatic(buffer, AL_FORMAT_MONO16, data, 4096, 44100);
```

6
```
// Create an OpenAL Source Object
alGenSources(1, &source);
```

7
```
// attach OpenAL Buffer to OpenAL Source
alSourcei(source, AL_BUFFER, buffer);
```

Setting up OpenAL

Creating sources and buffers

4
```
// Create an OpenAL Buffer Object to store audio data
alGenBuffers(1, &buffer);
```

5
```
// Fill the buffer with audio data
// Static buffer extension eliminates a buffer data copy
alBufferDataStatic(buffer, AL_FORMAT_MONO16, data, 4096, 44100);
```

6
```
// Create an OpenAL Source Object
alGenSources(1, &source);
```

7
```
// attach OpenAL Buffer to OpenAL Source
alSourcei(source, AL_BUFFER, buffer);
```

Setting up OpenAL

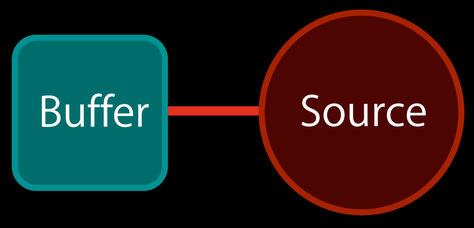
Creating sources and buffers

4
```
// Create an OpenAL Buffer Object to store audio data
alGenBuffers(1, &buffer);
```

5
```
// Fill the buffer with audio data
// Static buffer extension eliminates a buffer data copy
alBufferDataStatic(buffer, AL_FORMAT_MONO16, data, 4096, 44100);
```

6
```
// Create an OpenAL Source Object
alGenSources(1, &source);
```

7
```
// attach OpenAL Buffer to OpenAL Source
alSourcei(source, AL_BUFFER, buffer);
```



Setting up OpenAL

Play that tune!

```
// Set various attributes on the source
alSourcei (source, AL_LOOPING, AL_TRUE);
alSourcefv(source, AL_POSITION, source_position_xyz);
alSourcef (source, AL_REFERENCE_DISTANCE, 5.0f);
```

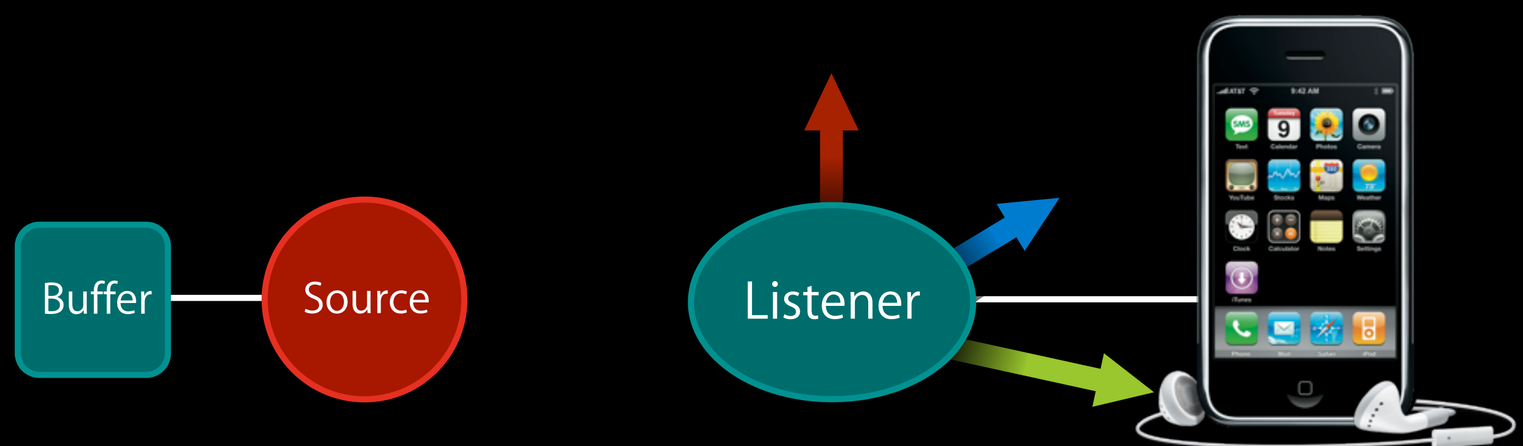
8

```
// Begin playing our audio source
alSourcePlay(source);
```

9

```
// then during gameplay...
alSourcefv(source, AL_POSITION, source_position_xyz);
alListenerfv(AL_POSITION, listener_position_xyz);
alListenerfv(AL_ORIENTATION, listener_orientation);
```

X



Setting up OpenAL

Play that tune!

8

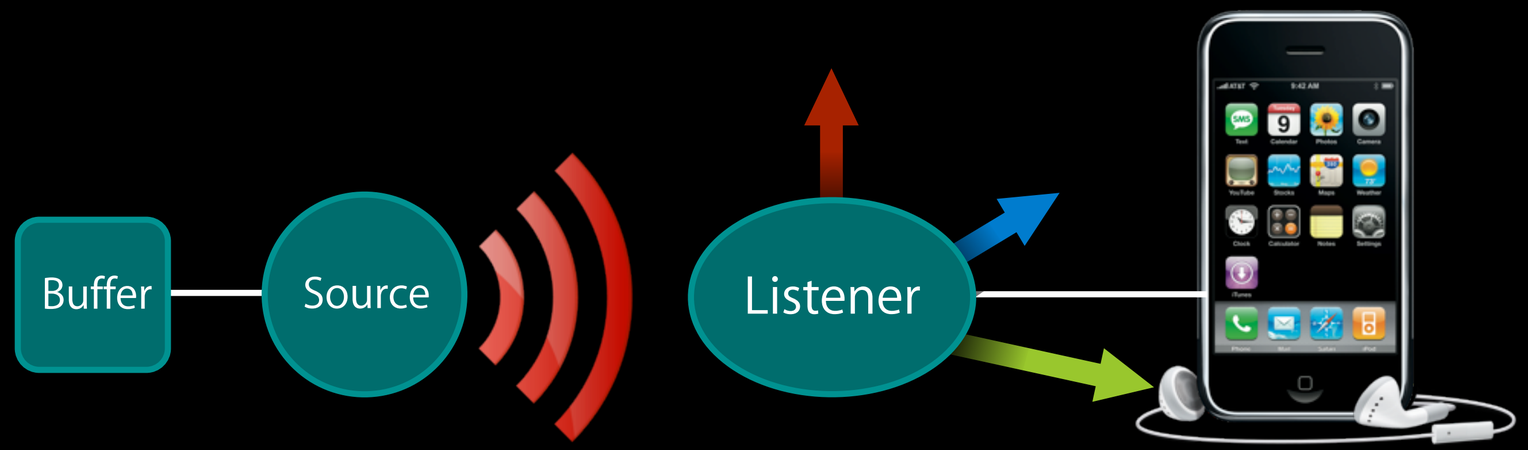
```
// Set various attributes on the source
alSourcei (source, AL_LOOPING, AL_TRUE);
alSourcefv(source, AL_POSITION, source_position_xyz);
alSourcef (source, AL_REFERENCE_DISTANCE, 5.0f);
```

9

```
// Begin playing our audio source
alSourcePlay(source);
```

X

```
// then during gameplay...
alSourcefv(source, AL_POSITION, source_position_xyz);
alListenerfv(AL_POSITION, listener_position_xyz);
alListenerfv(AL_ORIENTATION, listener_orientation);
```



Setting up OpenAL

Play that tune!

8

```
// Set various attributes on the source
alSourcei (source, AL_LOOPING, AL_TRUE);
alSourcefv(source, AL_POSITION, source_position_xyz);
alSourcef (source, AL_REFERENCE_DISTANCE, 5.0f);
```

9

```
// Begin playing our audio source
alSourcePlay(source);
```

X

```
// then during gameplay...
alSourcefv(source, AL_POSITION, source_position_xyz);
alListenerfv(AL_POSITION, listener_position_xyz);
alListenerfv(AL_ORIENTATION, listener_orientation);
```

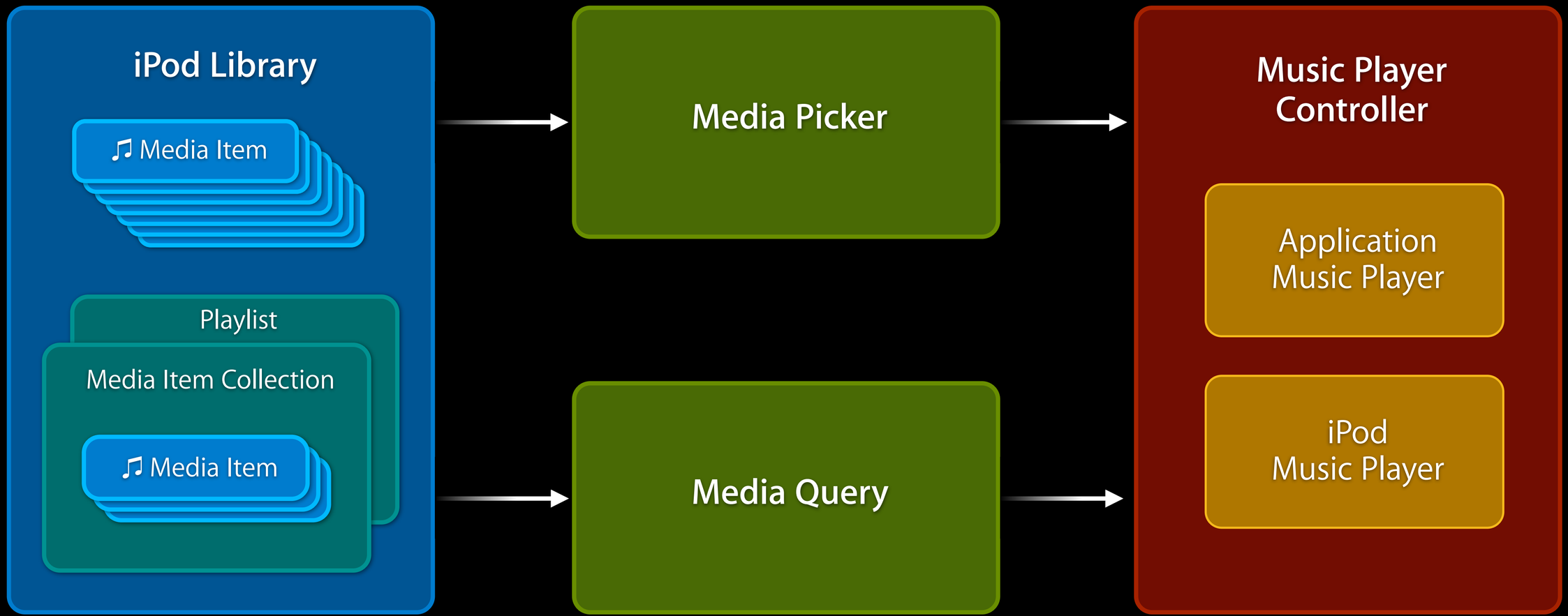
Accessing the iPod Library

iPod Library Access

Introduction

- Audio playback from iPod library
  - Play the user's music within your application
- Built-in UI for picking songs
- Perform searches, find playlists
- Mix playback with game audio

Media Player Overview



MPMediaItem

A single song, podcast, or audio book

♫ Media Item

Media Type	Music
Title	Hanging Around
Artist	The Venetian Blinds
Album	Open Up
Artwork	

…more…

MPMediaItem

A single song, podcast, or audio book

- Properties

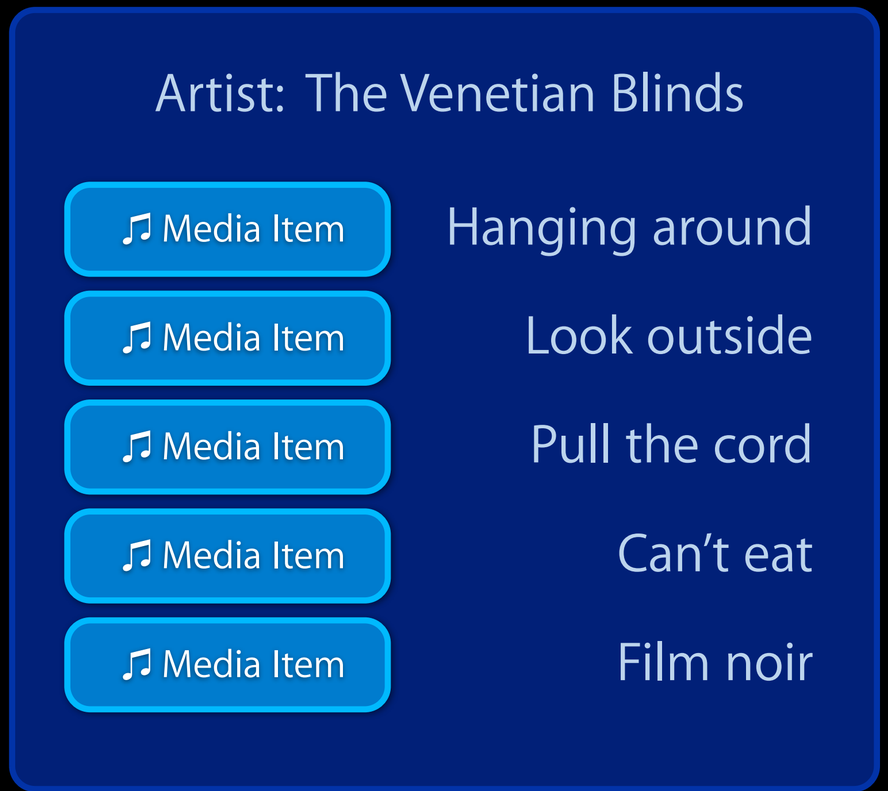
```
PersistentID
MediaType
Title
Artist
AlbumTitle
AlbumArtist
Genre
Composer
PodcastTitle
PlaybackDuration
AlbumTrackNumber
AlbumTrackCount
DiscNumber
DiscCount
Artwork
Lyrics
PlayCount
SkipCount
Rating
LastPlayedDate
```

- Retrieving values

```
- (id)valueForProperty:(NSString *)property;
```

MPMediaItemCollection

A collection of related media items



MPMediaItemCollection

A collection of related media items

- Ordered list of items

```
NSArray *items = [collection items];
```

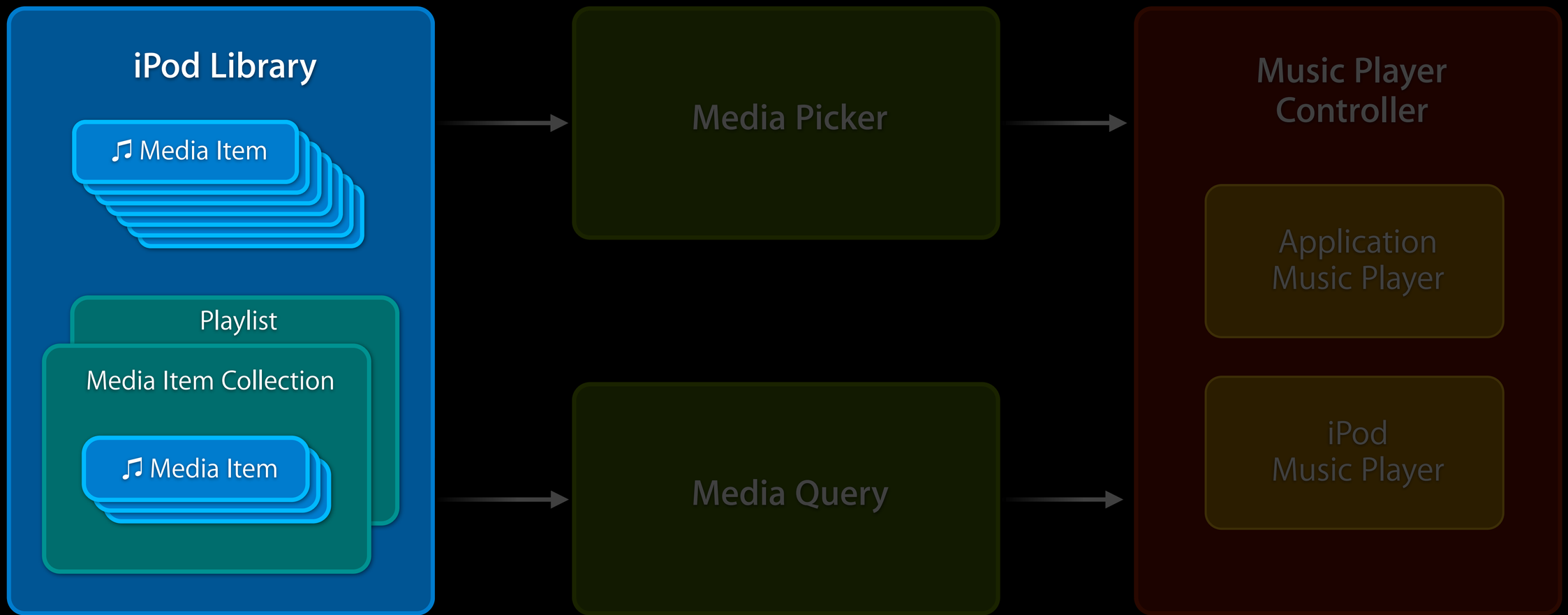
- Has a “representative item”

```
MPMediaItem *item = [collection representativeItem];
```

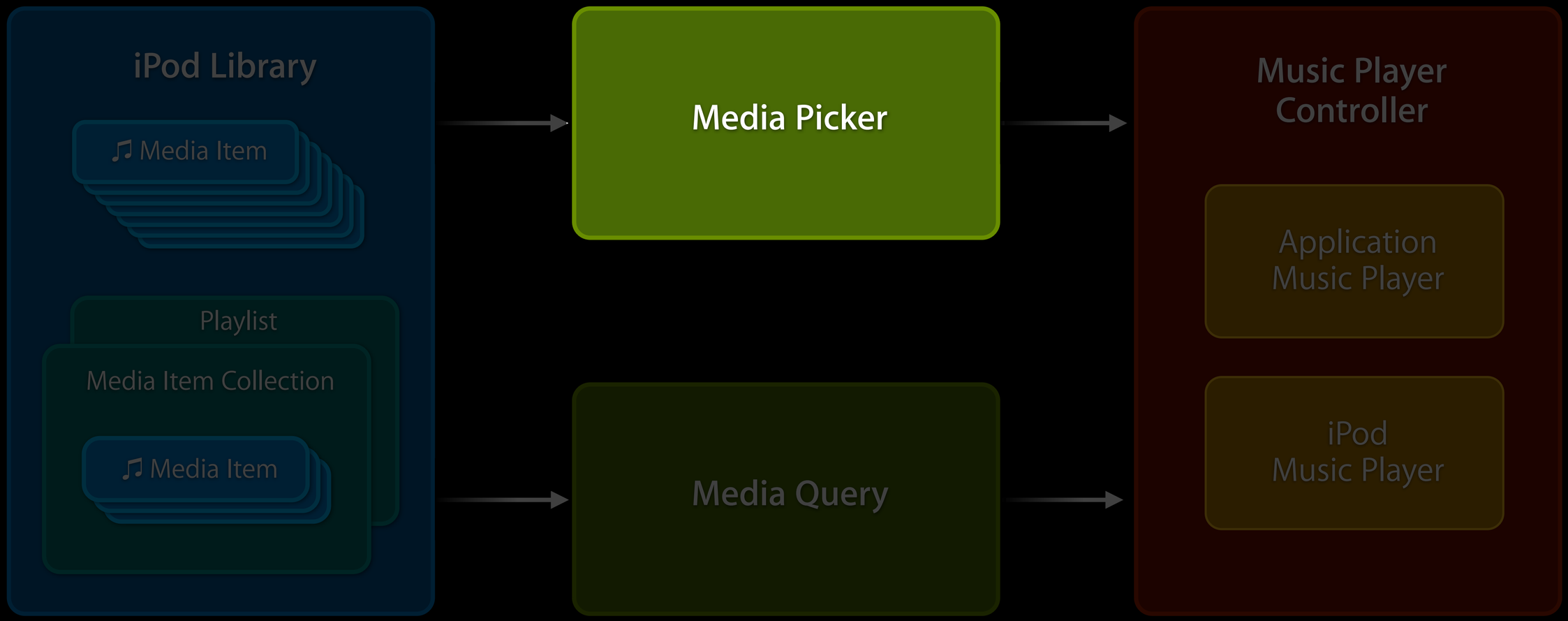
- MPMediaPlaylist subclass

```
MPMediaPlaylistPropertyName
MPMediaPlaylistPropertyPersistentID
MPMediaPlaylistPropertyPlaylistAttributes
MPMediaPlaylistPropertySeedItems
```

iPod Library

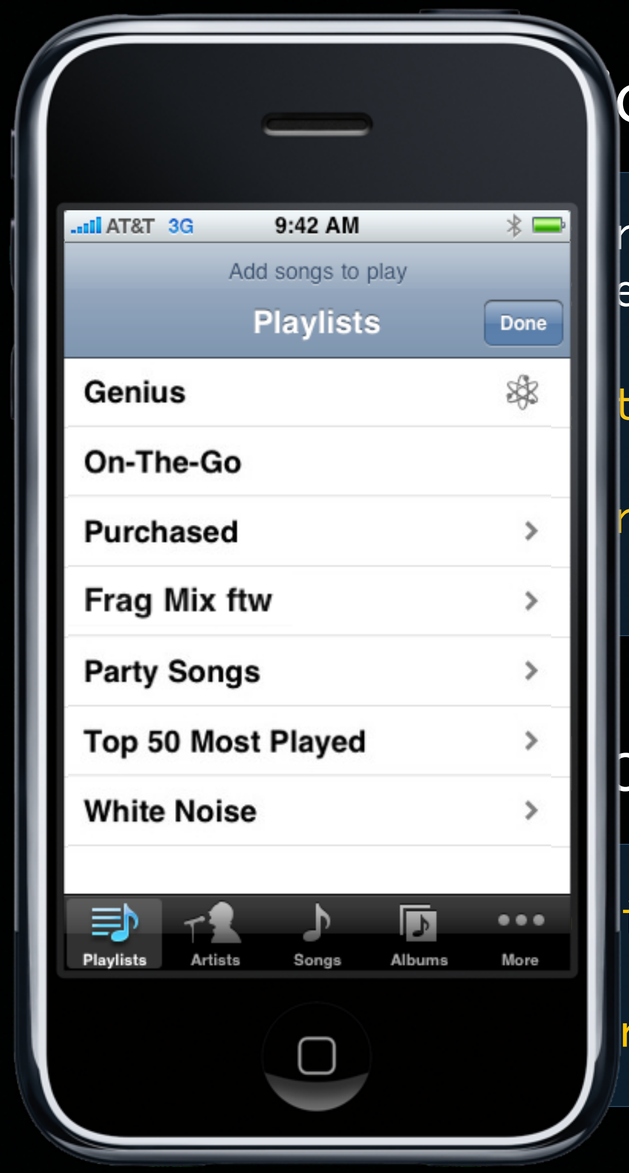


Selecting Media



MPMediaPickerController

Presents standard UI for picking media



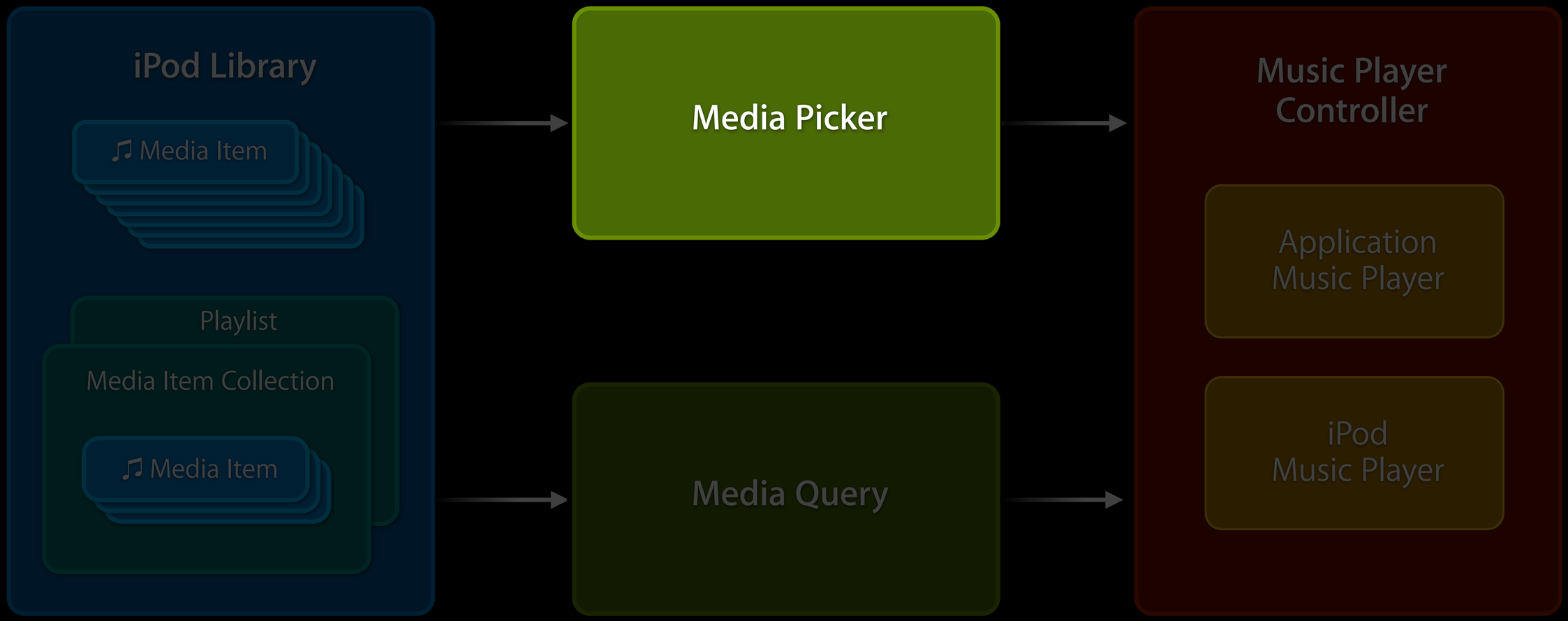
cker

```
roller *picker =
erController alloc] init];

te:self];

resentModalViewController:picker
                 animated:YES];
```

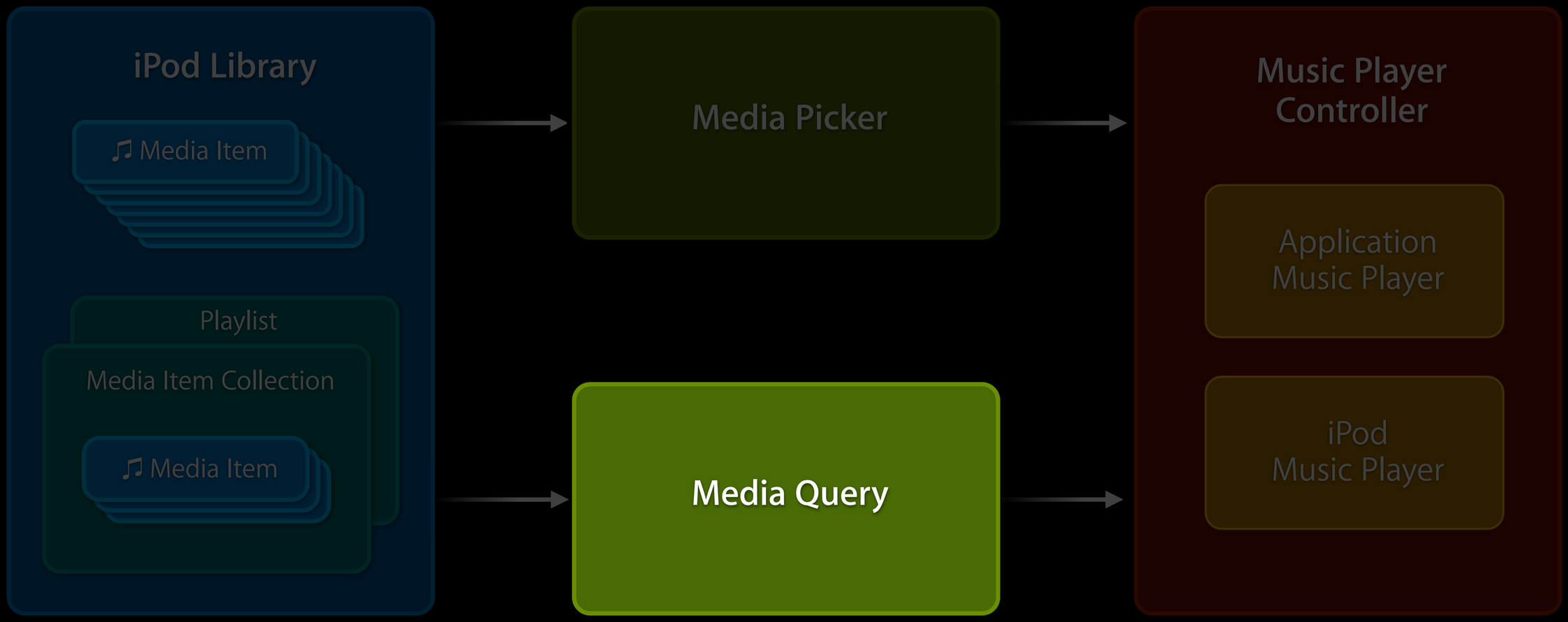
ods

```
ickMediaItems:

ncel:
```

Selecting Media



Selecting Media



MPMediaQuery

Search for items and collections

- Start from a built-in query

```
+albumsQuery          +playlistsQuery          +compilationsQuery
+artistsQuery         +podcastsQuery           +composersQuery
+songsQuery           +audiobooksQuery         +genresQuery
```

- Example

```
MPMediaQuery *query1 = [ MPMediaQuery songsQuery ];
NSArray *songs = [ query1 items ];                  // Result is all songs

MPMediaQuery *query2 = [ MPMediaQuery playlistsQuery ];
NSArray *collections = [ query2 collections ];      // Result is all playlists
```

MPMediaQuery

Search for items and collections

- Start from a built-in query

```
+albumsQuery            +playlistsQuery         +compilationsQuery
+artistsQuery           +podcastsQuery          +composersQuery
+songsQuery             +audiobooksQuery        +genresQuery
```

- Example

```
MPMediaQuery *query1 = [ MPMediaQuery songsQuery ];
NSArray *songs = [ query1 items ];                  // Result is all songs

MPMediaQuery *query2 = [ MPMediaQuery playlistsQuery ];
NSArray *collections = [ query2 collections ];      // Result is all playlists
```

MPMediaQuery

Search for items and collections

- Start from a built-in query

```
+albumsQuery            +playlistsQuery         +compilationsQuery
+artistsQuery           +podcastsQuery          +composersQuery
+songsQuery             +audiobooksQuery        +genresQuery
```

- Example

```
MPMediaQuery *query1 = [ MPMediaQuery songsQuery ];
NSArray *songs = [ query1 items ];                  // Result is all songs

MPMediaQuery *query2 = [ MPMediaQuery playlistsQuery ];
NSArray *collections = [ query2 collections ];      // Result is all playlists
```

MPMediaQuery

Search for items and collections

- Start from a built-in query

```
+albumsQuery          +playlistsQuery       +compilationsQuery
+artistsQuery         +podcastsQuery        +composersQuery
+songsQuery           +audiobooksQuery      +genresQuery
```

- Add Media Predicates to construct compound queries

```
+ (MPMediaPropertyPredicate *)predicateWithValue:(id)value
                                     forProperty:(NSString *)property
```

MPMediaQuery

Filtering by property

- Certain properties can be used in predicates

```
+[MPMediaItem canFilterByProperty:(NSString *)property];
```

- MPMediaItem

```
PersistentID            Artist              Genre
MediaType               AlbumTitle          Composer
Title                   AlbumArtist         PodcastTitle
```

- MPMediaPlaylist

```
PersistentID            Name                PlaylistAttributes
```

MPMediaQuery Examples

Compound playlist query

- Example

```
MPMediaQuery *query = [MPMediaQuery songsQuery];

// “Artist” property search
MPMediaPropertyPredicate *searchPredicate =
[ MPMediaPropertyPredicate predicateWithValue:@”The Venetian Blinds”
                                    forProperty:MPMediaItemPropertyArtist ];

// Add predicate to query
[ query addFilterPredicate: searchPredicate ];

NSArray *songs = [ query items ];    // All items matching the query
```

MPMediaQuery Examples

Compound playlist query

- Example

```
MPMediaQuery *query = [MPMediaQuery songsQuery];

// “Artist” property search
MPMediaPropertyPredicate *searchPredicate =
[ MPMediaPropertyPredicate predicateWithValue:@”The Venetian Blinds”
                                    forProperty:MPMediaItemPropertyArtist ];

// Add predicate to query
[ query addFilterPredicate: searchPredicate ];

NSArray *songs = [ query items ];    // All items matching the query
```

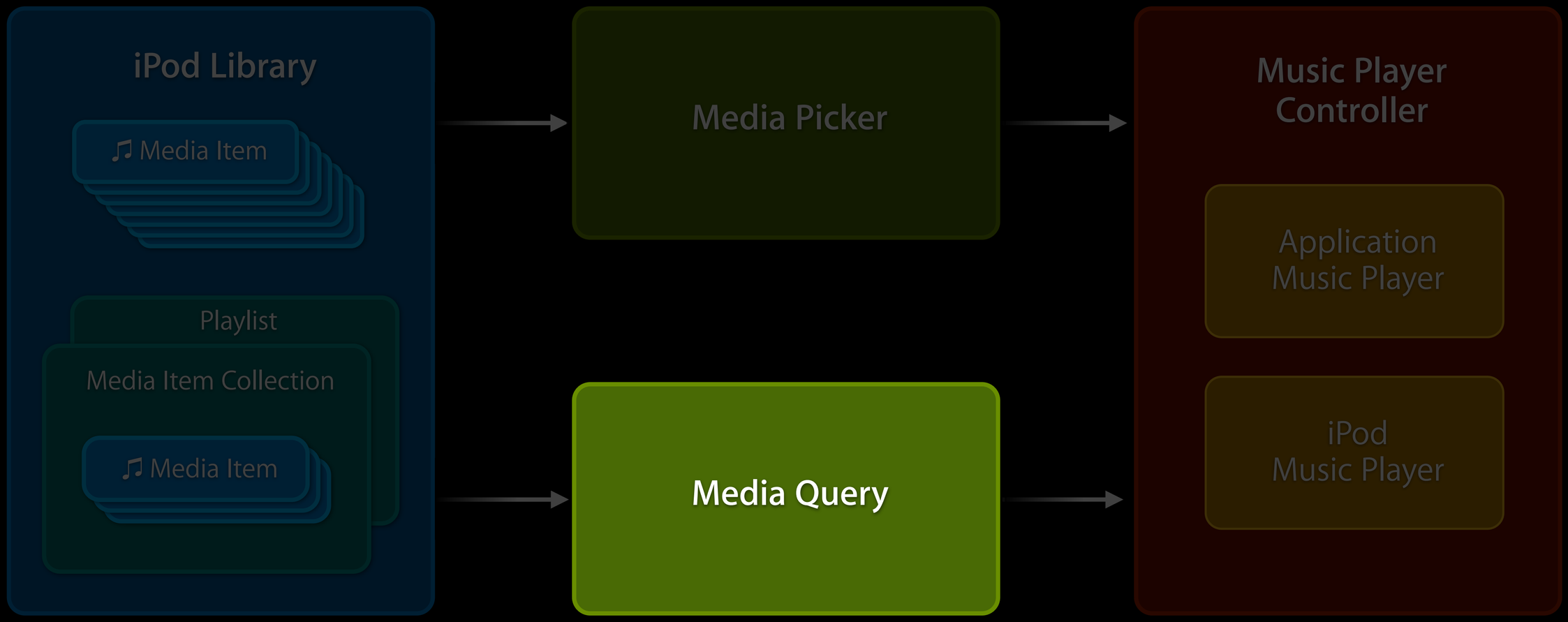
MPMediaQuery Examples

Compound playlist query

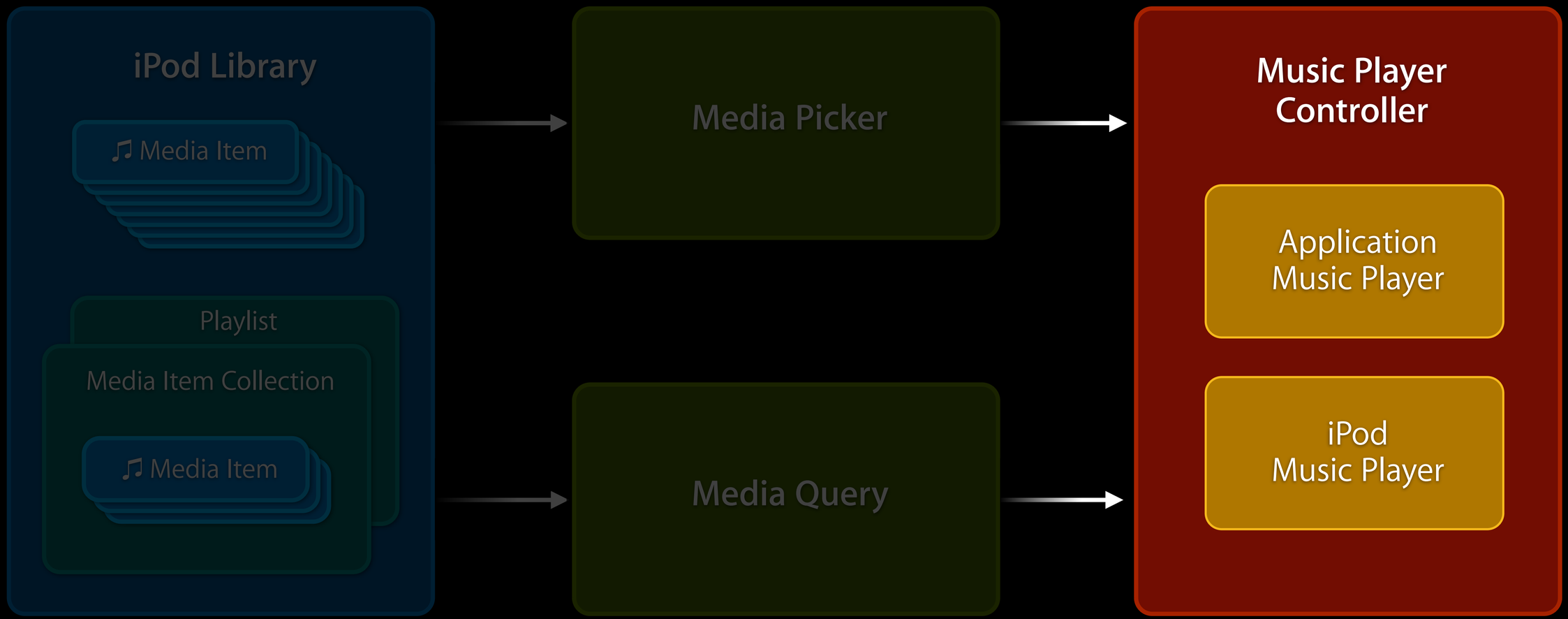
- Example

```
MPMediaQuery *query = [MPMediaQuery songsQuery];

// “Artist” property search
MPMediaPropertyPredicate *searchPredicate =
[ MPMediaPropertyPredicate predicateWithValue:@”The Venetian Blinds”
                                    forProperty:MPMediaItemPropertyArtist ];

// Add predicate to query
[ query addFilterPredicate: searchPredicate ];

NSArray *songs = [ query items ];    // All items matching the query
```

Selecting Media



Playing Media



MPMusicPlayerController

Plays items and collections

- Application Music Player
  - Music is played "within" your application
  - When your app quits, the music stops
- iPod Music Player
  - Music is played "from" the iPod player
  - Continues after your app quits

MPMusicPlayerController

Playing music

- Set up a queue of items

```
- (void)setQueueWithQuery:(MPMediaQuery *)query;

- (void)setQueueWithItemCollection:(MPMediaItemCollection *)collection;
```

- Set the playback mode

```
@property MPMusicRepeatMode repeatMode;   (Default, None, One, All)

@property MPMusicShuffleMode shuffleMode; (Default, Off, Songs, Albums)
```

MPMusicPlayerController

Controlling playback

- Start playback

```
- (void)play;

- (void)pause;

- (void)stop;
```

- Move through the items in the queue

```
- (void)skipToNextItem;

- (void)skipToPreviousItem;

- (void)skipToBeginning;
```

MPMusicPlayerController

Example

```
// Create a query (using previous example)
MPMediaQuery *query = ...

// Create and set up the music player
MPMusicPlayerController *musicPlayer =
    [ MPMusicPlayerController applicationMusicPlayer ];

// Set the player to endless repeat
myPlayer.repeatMode = MPMusicRepeatModeAll;

// Use the query to fill the player's queue
[ musicPlayer setQueueWithQuery:query ];

// Play that music!
[ musicPlayer play ];
```

MPMusicPlayerController

Example

```
// Create a query (using previous example)
MPMediaQuery *query = ...

// Create and set up the music player
MPMusicPlayerController *musicPlayer =
    [ MPMusicPlayerController applicationMusicPlayer ];

// Set the player to endless repeat
myPlayer.repeatMode = MPMusicRepeatModeAll;

// Use the query to fill the player's queue
[ musicPlayer setQueueWithQuery:query ];

// Play that music!
[ musicPlayer play ];
```

MPMusicPlayerController

Example

```
// Create a query (using previous example)
MPMediaQuery *query = ...

// Create and set up the music player
MPMusicPlayerController *musicPlayer =
    [ MPMusicPlayerController applicationMusicPlayer ];

// Set the player to endless repeat
myPlayer.repeatMode = MPMusicRepeatModeAll;

// Use the query to fill the player's queue
[ musicPlayer setQueueWithQuery:query ];

// Play that music!
[ musicPlayer play ];
```

MPMusicPlayerController

Example

```
- (void)mediaPicker:(MPMediaPickerController *)picker
      didPickMediaItems:(MPMediaItemCollection *)collection;
{
    // Create and set up the music player
    MPMusicPlayerController *musicPlayer =
        [ MPMusicPlayerController applicationMusicPlayer ];

    // Set the player to endless repeat
    myPlayer.repeatMode = MPMusicRepeatModeAll;

    // Use the query to fill the player's queue
    [ musicPlayer setQueueWithItemCollection:collection ];

    // Play that music!
    [ musicPlayer play ];
}
```

MPMusicPlayerController

Example

```
- (void)mediaPicker:(MPMediaPickerController *)picker
      didPickMediaItems:(MPMediaItemCollection *)collection;
{
    // Create and set up the music player
    MPMusicPlayerController *musicPlayer =
        [ MPMusicPlayerController applicationMusicPlayer ];

    // Set the player to endless repeat
    myPlayer.repeatMode = MPMusicRepeatModeAll;

    // Use the query to fill the player's queue
    [ musicPlayer setQueueWithItemCollection:collection ];

    // Play that music!
    [ musicPlayer play ];
}
```

AVAudioSession

Using AVAudioSession

Categories recommended for games

- Ambient
  - Appropriate for games that mix with other background music
    - Allows iPod audio playback
    - Not appropriate for games playing their own background music
  - Follows recommended behavior for games
    - Obeys ringer switch
    - Silences when screen locks

Using AVAudioSession

Categories recommended for games

- AmbientSolo
  - Appropriate for games that play their own background music
    - Interrupts iPod audio playback
    - Allows application access to audio hardware codec
  - Follows recommended behavior for games
    - Obeys ringer switch
    - Silences when screen locks

AVAudioSession

Setting your category

```
// Retrieve session instance
AVAudioSession *session = [ AVAudioSession sharedInstance ];

// Request the “Ambient” category
[ session setCategory:AVAudioSessionCategoryAmbient error:&errRet ];

// Set our session to be active
[ session setActive:TRUE error:&errRet ];

// Set up OpenAL
...

// Set up MPMediaPlayerController
...
```

AVAudioSession

Setting your category

```
// Retrieve session instance
AVAudioSession *session = [ AVAudioSession sharedInstance ];

// Request the “Ambient” category
[ session setCategory:AVAudioSessionCategoryAmbient error:&errRet ];

// Set our session to be active
[ session setActive:TRUE error:&errRet ];

// Set up OpenAL
...

// Set up MPMediaPlayerController
...
```

AVAudioSession

Setting your category

```
// Retrieve session instance
AVAudioSession *session = [ AVAudioSession sharedInstance ];

// Request the “Ambient” category
[ session setCategory:AVAudioSessionCategoryAmbient error:&errRet ];

// Set our session to be active
[ session setActive:TRUE error:&errRet ];

// Set up OpenAL
...

// Set up MPMediaPlayerController
...
```

AVAudioSession

Setting your category

```
// Retrieve session instance
AVAudioSession *session = [ AVAudioSession sharedInstance ];

// Request the “Ambient” category
[ session setCategory:AVAudioSessionCategoryAmbient error:&errRet ];

// Set our session to be active
[ session setActive:TRUE error:&errRet ];

// Set up OpenAL
...

// Set up MPMediaPlayerController
...
```

Going Further

- AVAudioSession
  - Additional categories
  - Handling of interruptions and state changes
  - Preferred hardware settings
- Audio Session Services
  - Additional flexibility and advanced behaviors

Audio Development for iPhone OS	Presidio Tuesday 3:30PM

Agenda

- Part 1
  - Welcome and Overview — Done
  - Graphics — Done
  - Audio — Done
- Part 2
  - Peer-to-Peer Connectivity — 20 minutes
  - Accelerometer — 20 minutes
  - In-App Purchase — 20 minutes

Agenda

- Part 1
  - Welcome and Overview — Done
  - Graphics — Done
  - Audio — Done
- Part 2
  - Peer-to-Peer Connectivity — 20 minutes
  - Accelerometer — 20 minutes
  - In-App Purchase — 20 minutes

Related Sessions

OpenGL ES Overview for iPhone OS	Presidio Wednesday 2:00PM
OpenGL ES Shading and Rendering	Presidio Wednesday 3:30PM
OpenGL ES Tuning and Techniques	Presidio Wednesday 5:00PM
Audio Development for iPhone OS	Presidio Tuesday 3:30PM
Audio Processing and Playback	Presidio Tuesday 5:00PM
Accessing the iPod Library	Pacific Heights Thursday 5:00PM

Lab

Game Development for iPhone Lab	iPhone Lab C Thursday 9:00AM

More Information

Allan Schaffer
Graphics Technology Evangelist
aschaffer@apple.com

Documentation
iPhone Application Programming Guide
http://developer.apple.com/iphone

Q&A

WWDC Session Survey

You can rate your sessions on the WWDC Attendee Site.

Survey forms are located on each session description page at:

developer.apple.com/wwdc/attendee

WWDC09